A Raycast Approach to Collision Avoidance in Sailing Robots

Sauze, Colin; Neal, Mark

Publication date:
2010

Citation for published version (APA):
Sauze, C., & Neal, M. (2010). *A Raycast Approach to Collision Avoidance in Sailing Robots*.
http://hdl.handle.net/2160/4680

A Raycast Approach to Collision Avoidance in Sailing Robots

Colin Sauze
Department of Computer Science
Aberystwyth University
Aberystwyth, Ceredigion, United Kingdom, SY23 3DB
Email: cjs06@aber.ac.uk

Mark Neal
Department of Computer Science
Aberystwyth University
Aberystwyth, Ceredigion, United Kingdom, SY23 3DB
Email: mjn@aber.ac.uk

***Abstract*—This paper presents a simple mechanism for an autonomous sailing robot to detect when it is within close proximity to fixed obstacles and a reactive mechanism to avoid those obstacles. This is achieved by using a raster based map of the local area and raycasting from the boat's current position in order to determine the distance and heading to the nearest coastline. Once this is determined a new target heading which does not result in any immediate danger is computed. Simulations have shown that it is possible for a robot to sailing between a set of waypoints while avoiding obstacles placed between those waypoints. This method has been shown to be capable of selecting a sensible course and avoiding collisions in most cases, however when the robot becomes trapped in small inlets or between groups of tightly packed islands it can fail to find a suitable course.**

I. Introduction

In order to be considered fully autonomous sailing robots must be able to avoid collisions both with fixed obstacles such as shorelines and moving obstacles such as other boats. Although the legal status of sailing robots is very vague at present it is likely that as their presence becomes more common that legislative bodies will create rules governing them and that these rules will include a provision requiring some kind of collision avoidance mechanism. Additionally any operator of a sailing robot will also wish to avoid collisions in order to avoid the loss of their robot and reduce the need to continuously monitor it.

Many algorithms already exist to facilitate obstacle avoidance in robotics. Many of these involve planning a path in advance and can safely assume that conditions will remain relatively stable between a robot detecting a potential collision with an obstacle and completing its movement to avoid that obstacle. However, sailing robots can be somewhat different as they cannot sail directly up wind they are at the mercy of the wind to determine where they can travel and any shifts in wind direction may rapidly render a previously plausible plan implausible. This can be especially true when sailing close to the coast or on inshore waters where local topography can produce dramatic shifts in wind direction when changing position by just a few metres. Therefore it would seem more logical to operate in a more reactive manner, simply making short term adjustments to the robot's course to avoid immediate danger. Reactive control systems are well known in other branches of robotics and will be discussed further in section III.

As processing power is often limited on sailing robots a computationally simplistic method is required to detect the presence of obstacles and to determine appropriate courses to avoid them. This problem can at least in part be solved simply by giving the robot a map of its local area and using GPS to determine its position on that map. The nearest coastline can then be detected using a ray casting mechanism. It would also be possible to overlay other information sources such as the location of storms or shipping onto this map and simply use the same algorithm to detect and avoid them. Once hazards have been identified then an a new course must be selected which avoids a collision with this obstacle and any other obstacles in the vicinity.

II. Motivations

This work is motivated by a desire to be able to operate sailing robots with a guarantee that they will not attempt to sail through a land mass in order to get from one waypoint to another. There is also an additional motivation to stimulate the discussion of how to perform collision avoidance amongst the Microtransat/WRSC/Sailbot community of sailing robot builders. Two very specific motivations originally inspired this work. The first of these originated during the 2007 Microtransat competition hosted in Aberystwyth. During this event each sailing robot was equipped with a GPS tracking device which transmitted the boat's position over a GSM network every few minutes. Originally its was expected that there would not be enough chase boats available for each robot to be allocated one. Therefore it was desirable to be able to automatically alert the chase boats if a robot was too close to the shore or to another robot. It was envisaged that the raycast algorithm described in this paper would be suitable for producing such warnings from the tracking data and a map of hazardous areas. Eventually sufficient chase boats were found and the need for this system diminished and it was never fully implemented. The second motivation was as part of research into biologically inspired neuro-endocrine control systems. One of the aims of this control system was to balance long term needs of the robot between performing its mission, keeping the batteries charged and avoiding danger. The ability to avoid collisions fulfils a large part of the

need to avoid danger and provides an ideal example which should be sufficient to demonstrate the capabilities of the neuro endocrine controller.

III. Background

Traditionally there have been two schools of thought regarding collision avoidance techniques in robotics. Deliberative methods work by using a model of their world and planning a path through that world, they rely on all necessary information to plan that path being available to them. Should the system discover that the world has changed then the model of the world must be updated and the path re-planned. Typically to avoid collisions deliberative robots will attempt to build a model of the world from sensor data and plan a route through that model. An early example of such a robot was the SHAKEY robot [1] built at Stanford University in the late 1960s, this used a laser range finder to build models of a room and was capable of rebuilding its model should it encounter an previously unknown object. These methods were further advanced in the 1980s by the work of Borenstein and Koren [2] who attempted to classify obstacles into moving and non-moving and created methods to ensure that a moving mobile robot would plan a path that would not collide with other moving obstacles. They later devised the idea of a "Virtual Force Field" [3] which is placed around obstacles to influence the planned path away from those obstacles.

Reactive methods operate by simply reacting to the information immediately available to them through the robot's sensors'. This means that the robot is not able to see the "bigger picture" but that it does not need to build any model of the world and instead uses the world as its own model as it perceives through its sensors. Reactive robots use their sensors to detect when they are within close proximity to an obstacle and will then move until they are no longer in close proximity to any obstacles and then continue on their way. This can lead to problems of becoming stuck by local minima such as corners of a room. A common example of such reactive robots are Braitenberg Vehicles [4], these typically have 2 sensors which detect obstacle proximity (such as sonars, IR range finders or bump sensors) and two motors one on each side of the robot, when a stimulus is detected by a sensor the motor on the opposite side of the robot is slowed causing it to turn away from the obstacle.

Realising that both methods have their limitations attempts have been made to bridge the gap and employ reactive techniques as an immediate reaction to colliding or nearly colliding with an obstacle while deliberative techniques provide global path planning and longer distance navigation. These are often combined using techniques such as Brooks' Subsubmption Architecture [5] or Arkin's Schema approach [6].

A. Existing Approaches to Collision Avoidance in Non-Wheeled Robotics

More recently many researchers have attempted to apply a variety of general mobile robot collision avoidance techniques to powered autonomous surface craft or to act as a navigation aid on manned vessels. To the best of the author's knowledge nobody has tried with sailing robots. In 2000 Smierzchalski and Michaelwicz [7] demonstrated a method for ships to avoid other ships using an evolutionary strategy, their system required approximately 800 generations of evolution to generate a solution, this could be computed in under 1 minute and could track up to 20 ships. Although this method is clearly capable something with faster computation time would be ideal as it is quite possible that more than 20 ships are in the vicinity of a robot and the processing time maybe significantly higher on a small embedded computer as might be used in a sailing robot. In 2004 Lee and Kim [8] attempted to implement a subset of the International Rules for the Prevention of Collisions at Sea (COLREGs) in a fuzzy logic system. They focused on those rules which are concerned with making manoeuvres to avoid a collision and were able to demonstrate movements that are compliant with COLREGs. In 2006 Benjamin, Leonard et al [9], [10] developed a more extensive and complete implementation of an autonomous COLREGs system and demonstrated it using several powered autonomous kayaks. As with Lee and Kim's system they were not able to address all of the COLREGs as many of these concern tasks which are still only really achievable by humans such as maintaining a constant watch, responding to VHF radio messages or responding to vessels in distress. In 2009 Bandyophadyay, Sarcione and Hover [11] tested a reactive collision avoidance system onboard in Singapore harbour based around the same kind of kayak as Benjamin and Leonard. They used a laser scanner with a 250m range to detect obstacles, however this suffered from noise especially against large waves. They were able to successfully autonomously steer around a moored boat. As their system worked in a reactive manner based upon laser scanner data there is no attempt to distinguish the nature of the obstacle and this system should operate equally well (assuming the laser scanner has no problems sensing the obstacle) when avoiding a coastline, a ship or any other kind of obstacle. However the system does not take account of objects which might be moving as all of the other systems discussed here do.

Collision avoidance techniques used in other branches of outdoor robotics may also be of use in sailing robots. In particular systems used in unmanned aerial vehicles (UAVs) face many similar problems to autonomous surface craft (except they have to worry about 3 dimensions not 2) as they cannot simply stop or immediately reverse direction and are easily affected by wind. Of particular interest is work by Viquerat, Blackhall et al [12] who have developed a UAV with a forward facing doppler RADAR system that is able to detect obstacles 10-15 metres away at a rate of 10hz. Given this short range they opted for a reactive algorithm which avoids obstacles detected with the RADAR. They also note that deliberative path planning approaches suffer from increasing computational complexity as the number of obstacles increase while reactive methods do not, given the speed at which they must compute new trajectories it is vital to minimise computation time.

IV. Methodologies

A. Reactive Architectures

Given the ever changing nature of a sailing robot's environment it would seem more sensible to follow the reactive approach to collision avoidance. Any attempt to plan a path over any significant distance must take the wind direction into account. Especially when sailing in coastal or inshore waters it is difficult to determine wind direction in advance, therefore rendering any previously planned path invalid. However some higher level planning is needed, it is expected that the robot's operator has given the robot a predefined set of waypoints that they wish it to sail between and that these are reasonably sensible and that if a line were drawn between these they would not intersect major obstacles.

B. Raycasting

A technique known as raycasting has been selected to determine the heading and distance to an obstacle and to discover obstacle free routes to sail. Raycasting is a technique often used in early 3D computer games such as Wolfenstein 3D or Doom. It works by tracing the path of a series of rays originating from the players current position on a map of objects. When one of these rays hits an object (such as a wall) the distance is registered and the object is rendered at a size proportional to its distance. This can be applied to a robot by using the robot's GPS position to place the robot on a map and then casting the rays from the robot's position until they reach an obstacle (such as a land mass) on the map. This will result in the robot knowing the distance to the coastline in every direction. Based upon this information the robot can take appropriate action to avoid a collision with the coastline. A pseudo code algorithm is shown below.

```
for angle=0 ; angle<360 ; angle++
    for dist=0 ; dist<max_dist ; dist++
        x=sin(angle)*dist
        y=cos(angle)*dist

        if getpixel(x,y) not = 0
           if dist < nearest_dist
               nearest_x=x
               nearest_y=y
               nearest_dist=dist
```

Figure 1 shows an example raycast set against a map with some islands as obstacles.

Depending on the intended usage of the data there may not be a need to scan through a full 360 degrees as obstacles behind the robot may not be of concern. Instead a "beam" of 60 degrees centred around the current heading of the boat (As shown in figure 2) would be sufficient to detect any obstacle that the boat might sail into. However if the boat was not sailing properly or being dragged by tides or currents then this may not be a sensible option, one possible alternative is to derive the heading from GPS data instead of the compass

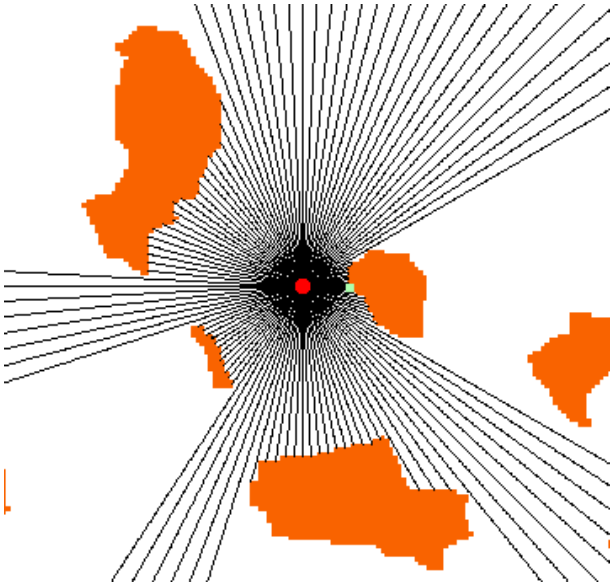

Fig. 1. An example raycast. The rays can be seem emerging from the boat which is represented by the circular dot. The square just to the right of this indicates the closest coastline to the boat. The obstacles are islands on the map.

heading as this will determine the direction the boat is actually travelling in, not the direction it is facing. If the beam is to be less than 360 degrees then an optimal with needs to be determined. Wider beams will detect obstacles which are not directly ahead of the robot but which might still be a collision risk if the robot alters course (either willingly or unwillingly). This could result in false positives being generated when obstacles appear at the extremities of the beam. Conversely a narrow beam will miss these false positives but will be more susceptible to missing actual obstacles if the boat changes course.

C. Obstacle Avoidance Strategies

Once an obstacle has been detected then appropriate action needs to be taken to avoid it. The avoidance strategy must select a new course which avoids colliding with the detected obstacle or any other obstacle.

1) Behaviour Switching: The robot needs to eventually return to sailing towards its waypoint. Once a new course has been decided upon that course (or one very close to it) must be followed for enough time to allow the robot to actually complete the change course and sail far enough to avoid the collision. There is a need to commit to the avoiding action and stay with it for sufficient time that it becomes useful. Conversely this time cannot be too long or the robot may become at risk of colliding with another obstacle that was not previously accounted for. Part of the author's other research is concerned with biologically inspired behaviour switching and action selection mechanisms based upon an abstracted version of the neural and endocrine systems. In these systems, chemical messengers known as hormones are released in response to certain stimuli, these then bind with

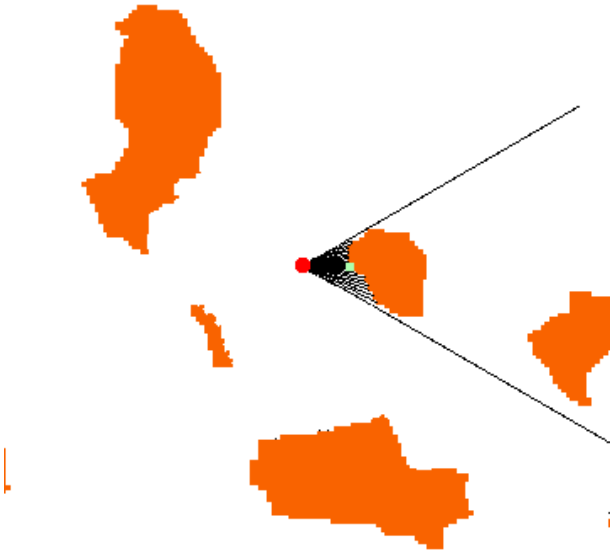

Fig. 2. An raycast with the beam limited to a 60 degree width, centred around the current heading of the robot.

receptors on target cells and trigger a change in the behaviour of those cells. These changes can take place on timescales varying between a few seconds and many months, the result can be that behavioural changes occur gradually with multiple behaviours being simultaneously exhibited. Such properties could be useful in a collision avoidance system to allow a gradual transition between sailing a course which avoids an obstacle and one which sails towards a pre-designated waypoint. However a gradual switching between sailing towards the waypoint and avoiding an obstacle (when the obstacle is first detected) may not be as desirable as an early and decisive action is more likely to avoid a collision than a gradual one which leaves things too late.

To achieve these properties when an obstacle is detected hormone is not released directly but instead it is stored (also known as pooling) until a threshold value is reached, then all of the stored hormone is released at once triggering a sudden change in behaviour. This hormone then gradually decays to return to a normal behaviour. This is achieved by setting the heading that the robot follows to be determined by multiplying the hormone quantity (which is always between 0 and 1) by the difference between the robot's original heading and the new heading that was determined by the obstacle avoider. As the hormone decays the target heading gradually returns to its original value. The end result is for the first few seconds when the robot approaches an obstacle nothing happens while the hormone pools, then the hormone is released and there is a dramatic change in heading and then the robot gradually returns to its original heading. If the robot is still in danger of collision then the process will repeat itself, it is vital that the hormone decay is quite slow to allow enough time for the obstacle to be passed and it is also vital that the initial hormone release is fast enough to allow action to be taken before a collision occurs.

2) Deciding on the new course: The first attempt at an algorithm to avoid collisions was simply 180 degrees away from the obstacle, this worked fine for simple round obstacles and where no other obstacles where present nearby, but when more complex coastlines with jagged edges where used the robot often oscillated between two headings. At best this caused it to hold station in a small area, at worst the momentum lost during the turn resulted in a collision. At second attempt had the robot sail towards the heading with the longest clear path, this prevented it becoming stuck by inlets but often caused the robot to sail away from its intended destination, on other occasions it would constantly change direction as the heading with the longest clear path could potentially change by 180 degrees from one iteration to the next.

The final algorithm to find the new course searches for a clear course to sail that is as close as possible to the current target heading (rather than the actual course). The reason for this is to minimise the change in course in order to keep the boat going in the correct general direction and to minimise rudder/sail movements which are relatively expensive in terms of power consumption. To achieve this the algorithm checks the distance to the nearest coastline by alternating either side of the target heading, the first course which is found to be safely sailable is selected. For these simulations a distance of 150 metres was selected. An additional criteria that the new course must differ from the current course by at least 5 degrees as courses of less than 5 degrees don't actually trigger a change in rudder position in our control system. This strategy worked for the most part but it was observed that the selected course often barely cleared the obstacle and that sometimes even resulted in a collision. So an additional rule was added that the selected heading must also be at least 5 degrees away from the nearest point where the coast was of a distance less than 150 metres. This does leave the obvious limitation that if the robot becomes trapped inside an inlet where they only way out is through a narrow passage that is less than 5 degrees wide then it will never be able to escape from the inlet.

Figure 3 illustrates an example of the heading selection algorithm split into seven stages. The desired heading is shown as a solid line with an arrow, the rays which are checking the distance to the coast are shown as dashed lines. In stages one to five the rays are always finding that the coastline is within 150 metres so these cannot be used as valid courses. In stage 6 a ray goes beyond 150 metres but it is still within 5 degrees (angles in this diagram have been exaggerated for illustrative purposes). Finally in stage 7 a ray goes beyond 150 metres and is more than 5 degrees clear of the coastline, so this is selected as the new heading.

3) Dealing with tacking: Additional problems are presented when sailing up wind. Sailing boats cannot sail directly into the wind, sailors often quote the figure of 45 degrees as the minimum angle between the boat's heading and the wind, although this figure can vary depending on the boat. This is often referred to as the "no go zone". This limits the number of potential courses that the robot can sail to avoid a collision as

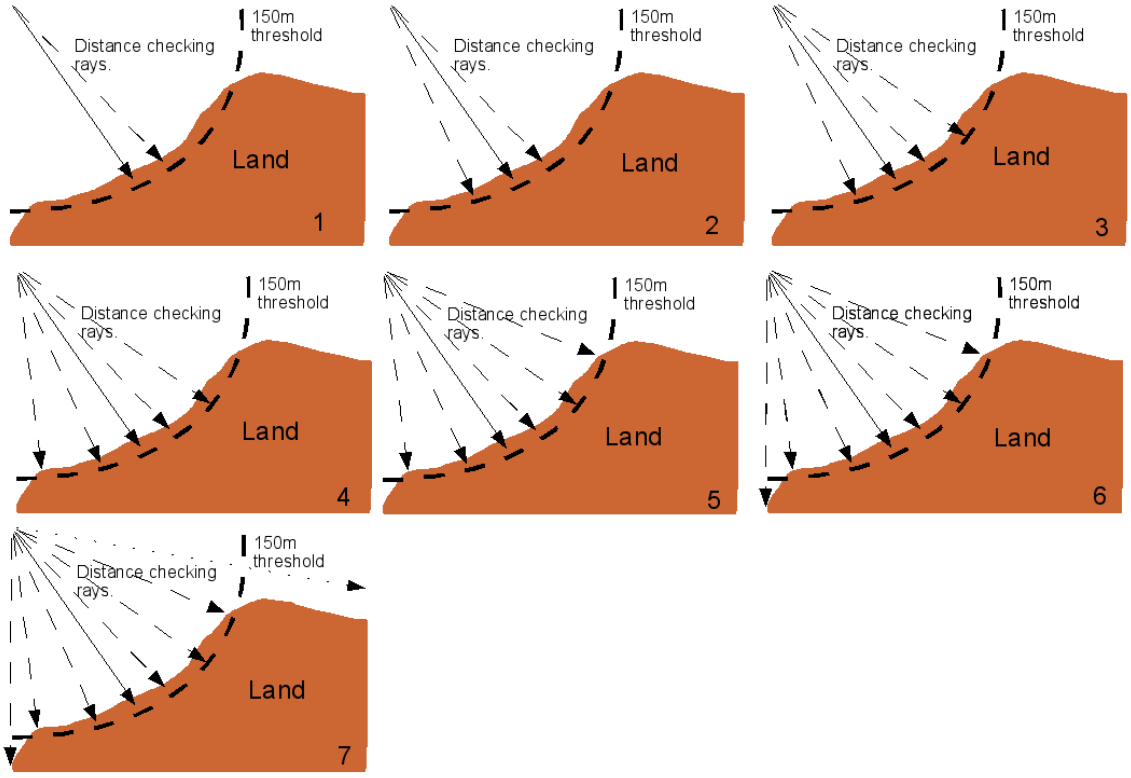

Fig. 3. The heading selection algorithm.

the no go zone must be ruled out. So the alternating algorithm previously described must be modified to avoid searching in the no go zone.

V. Results

The work presented in this paper has been simulated using a modified version of the tracksail-AI program [1]. The simulations where conducted using a map of Strangford Lough in Northern Ireland (54.5 degrees North, 5.6 degrees West - see figure 4). It is the largest inlet in the British Isles at $150km^2$ in area, approximately 30km long at its longest point and contains over 70 islands (although many of these are often submerged by the tide). This was chosen as it presents a very challenging environment to sail in with plenty of small inlets and groups of islands which could easily trap a robot and confuse collision avoidance algorithms. A map based on the OpenStreetMap map was modified (as shown in figure 5 to remove all land features and only use two colours, one of the water and one for the land.

An experiment was devised to sail between two waypoints when the direct line between them was clearly obstructed. The algorithm would therefore be required to avoid sailing through the land and find a suitable path. The route that it took is shown in figure 5, the start point is labelled “start“ and the

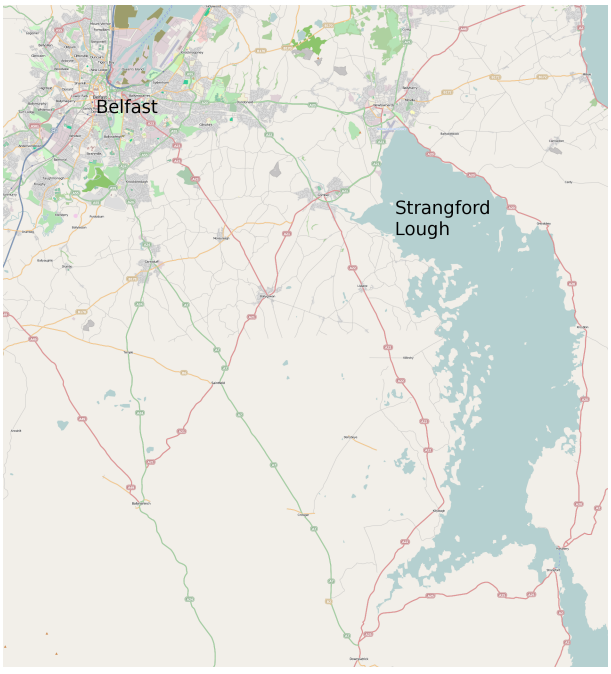

Fig. 4. A map of Strangford Lough from OpenStreetMap.

[1] http://microtransat.svn.sourceforge.net/viewvc/microtransat/tracksail-AI/

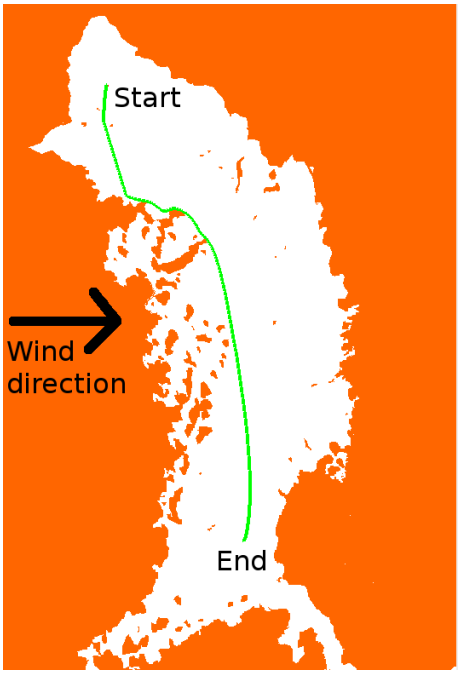

Fig. 5. The course sailed by the robot in Strangford Lough. The only waypoints specified are those at the points labelled start and end.

end point ”end“, no other waypoints where specified. The wind was set to be consistently from the West which did allow the robot to sail on a beam reach which is usually considered the fastest, most stable and easiest point of sail to use. A further challenge for future work is to setup a triangular course where the collision avoidance algorithm must cope with wind from all directions.

A number of attempts where made to get the boat to navigate through the narrow channel at the south of Strangford Lough which leads into the Irish sea. However because this channel is so narrow the boat would often fail to see any possible sailable route and would just continue sailing its existing course causing a collision. This has in part been made worse by an implementation decision in the simulator, the translation between units of distance in the real Strangford Lough and the simulated one are somewhat arbitrary and the simulator believes this channel to be less than 100 metres wide whereas in reality it is over 500m wide at its narrowest point. The system needs to be modified so that if a solution meeting the criteria specified in section IV-C2 cannot be found that the nearest possible solution is returned instead of giving no solution and sailing into the shore! There are also several points where the robot appears to almost clip the coastline and sails very close despite the rule requiring 5 degrees of clearance, perhaps a better strategy would be to require a suitable course to achieve a minimum distance rather than a minimum angle.

A. Dealing with local minima

Figure 6 shows an example of how the boat can become trapped in inlets. In this example the boat was supposed to

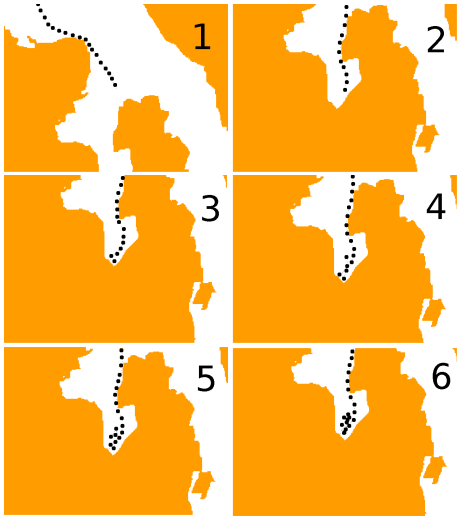

Fig. 6. An example of the algorithm becoming stuck inside a small inlet.

sail down the channel on the right but instead chose to sail to the left to avoid a collision with the peninsula in the centre. It was then unable to escape from the inlet because it detected a potential collision as it nearer the entrance to the inlet. A narrower beam width might have allowed it to proceed but may have caused problems elsewhere. Some potential solutions to this are to either modify the beam width when the boat becomes trapped, to set narrow inlets as no go zone's on the map so that the robot will not view them as a potentially valid place to sail (this will require all maps to be checked before use) or to attempt to use an edge following routine to just follow the coastline until open water is found.

VI. Conclusions and Future Work

A. Using a real robot

If time permits this experiment will be recreated on a real robot which will avoid some buoys that have been placed in its way. Unfortunately the winter of 2009/2010 (when much of this work has been undertaken) in Wales has not been favourable for testing sailing robots and what experiment time that has been available had already been allocated to other experiments. Hopefully spring will be bring better weather and additional results will be available for the final version of this paper and a full demonstration of this algorithm will be shown at the 2010 World Robotic Sailing Championships in Kingston, Ontario.

B. Limitations

There are still several key limitations to this work. Firstly no account of the tides or currents are made and these should be taken into account when deciding upon an appropriate course to sail in order to avoid an obstacle. This would be especially true if really sailing in Strangford Lough as the narrow entrance to the Lough creates very strong tidal currents.

The ability to avoid obstacles using this system is only as good as the data provided in the map files. An ideal data source

would include submerged or semi-submerged obstacles. There may be some need for reprocessing of data depending on the draught of the boat it is to be used with. Potential data sources are discussed in the next section.

As shown in section V-A the boat can become stuck inside inlets or between groups of islands. In these cases it may be necessary to have a higher level path planning system which can produce a complete plan to get back to clear water. Such situations might be avoided (or at least reduced) by creating additional waypoints along straight lines between the normal waypoints, this would reduce the opportunity for the robot to "wander" off course between waypoints. Having prespecified waypoints which clearly avoid danger areas should always be preferable to relying on an autonomous routing system to avoid those areas.

C. Global coastline databases

A sailing robot could be loaded with a database of the entire global coastline could generate a map of its surrounding area for use with the raycaster. This could allow the robot to sail anywhere in the world and be able to avoid collisions with the coastline. Of course this assumes that the database of the coastline is sufficiently accurate, up to date and it will not include moving hazards such as ice. An ideal database would also include submerged obstacles such as shipwrecks, sand bars and rocks and would outline the coastline at the low water mark. Several potential data sources are presented below, although none of them represent a totally ideal solution. A few potential sources are listed below.

- PGS (Prototype Global Shoreline) [2]- A free to use database of global coastlines produced by the US government. Resolution is quite poor and there are explicit warnings on the webpage that it is not suitable for navigation, although it may be sufficient if the aim is to always remain 10s of kilometres from shore.
- DNC (Digital Nautical Charts) [3] - These digital charts are produced by the US government and are listed are suitable for navigation, but their usage is restricted outside the USA.
- GSHHS (Global Self-Consistent Hierarchical High resolution Shoreline) [4] - A high resolution shoreline dataset based on the combination of two previously available public sources.
- STRM (Shuttle Radar Topography Mission) [5] - A digital elevation model of the world taken from a RADAR on board the space shuttle.
- OpenStreeetMap [6] - This is an open source project aiming to produce an open source free to use map of the entire world. Its coastline data is constructed from a number of free sources including PGS and STRM data. However the accuracy can vary depending on the original data source. There may also be inconsistencies between whether the high water, mean water or low water mark was used to establish the coastline position.

These databases are not particularly small but could realistically be placed on a computer operating a sailing robot. For example the entire OpenStreetMap dataset is 160 gigabytes uncompressed (this includes data irrelevant to nautical use such as streets, topography, city names etc). When compressed or reduced to just coastline data this database could easily be placed on a flash memory device stored inside the robot. It is not inconceivable that even a small microcontroller connected to a flash memory device could access and process this data.

D. Adding obstacles other than coastline

There is no reason that the methods presented in this paper need to be restricted to avoiding only fixed obstacles. A composite map including coastline data, ship locations from Automatic Identification System (AIS) and weather information could be created. This would allow the robot to avoid collisions with any of these hazards. Instead of plotting ships simply as dots on the map they could be plotted as cones who's length is proportional to the ship's speed and who's direction is that of the ship's direction of travel.

E. Compliance with COLREGs

Ideally a collision avoidance system for a sailing robot that can avoid other shipping will do so in a manner which is compliant with the International Rules for Preventions of Collisions at Sea (COLREGs) as was implemented by Benjamin, Leonard et al [9], [10] and Lee and Kim [8]. To be of serious use the methods presented in this paper need to be combined with some kind of COLREGs rule checking system which could ensure that movements taken to avoid collisions with vessels are done in a COLREGs compliant manner.

F. Using RADAR

It is worth noting that the data resulting from the raycast method demonstrated in this paper bear a very strong resemblance to those which would be returned by a RADAR. Therefore there should be no major reasons stopping these algorithms from being applied to RADAR data almost without modification. A boat equipped with a RADAR would be able to use this algorithm to travel around without any map data and avoid collisions with any obstacle that will register on RADAR. These techniques could also be applied to a laser scanner as used by Bandyophadyay, Sarcione and Hover [11].

[2]http://www.nga.mil/portal/site/nga01/index.jsp?epi-content=GENERIC&itemID=9328fbd8dcc4a010VgnVCMServer3c02010aRCRD&beanID=1629630080&viewID=Article

[3]http://www.nga.mil/portal/site/dnc/

[4]http://www.ngdc.noaa.gov/mgg/shorelines/gshhs.html

[5]http://www2.jpl.nasa.gov/srtm/

[6]http://www.openstreetmap.org

REFERENCES

[1] N. J. Nilsson, "A mobile automation: An application of artificial intelligence techniques," in *Proceedings of the IJCAI*, 1969. [Online]. Available: http://www.ai.sri.com/pubs/files/1302.pdf

[2] J. Borenstein and Y. Koren, "Optimal path algorithms for autonomous vehicles," *CIRP Manufacturing System*, vol. 16, no. 4, pp. 297–309, 1987. [Online]. Available: http://www-personal.umich.edu/~johannb/Papers/paper04.pdf

[3] ——, “Real-time obstacle avoidance for fast mobile robots,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.22.8555

[4] V. Braitenberg, *Vehicles - Experiments in Synthetic Physcology*. MIT Press, 1996, no. ISBN 0-262-52112-1.

[5] R. Brooks, “A robust layered control system for a mobile robot,” *IEEE Journal of Robotics and Automation*, vol. 2, pp. 14–23, 1986. [Online]. Available: http://people.csail.mit.edu/brooks/papers/AIM-864.pdf

[6] R. C. Arkin, “Homeostatic control for a mobile robot: Dynamic replanning in hazardous environments,” *Journal of Robotic Systems*, vol. 9, pp. 197–214, 1992.

[7] R. Smierzchalski and Z. Michalewicz, “Modeling of ship trajectory in collision situations by an evolutionary algorithm,” *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 227–241, 2000. [Online]. Available: http://www.cs.adelaide.edu.au/~zbyszek/Papers/p41.pdf

[8] Y.-I. Lee and Y.-G. Kim, *A Collision Avoidance System for Autonomous Ship Using Fuzzy Relational Products and COLREGs*. Springer Berlin / Heidelberg, 2004, vol. 3177/2004, no. ISBN 978-3-540-22881-3. [Online]. Available: http://www.springerlink.com/index/c62fju712fp9f10h.pdf

[9] J. C. M. R. Benjamin, J.J. Leonard and P. M. . Newman, “A method for protocol-based collision avoidance between autonomous marine surface craft,” *Journal of Field Robotics*, vol. 23, no. 5, pp. 333–346, April 2006 2006. [Online]. Available: http://maribotics.com/publications/9_jfr-paper.pdf

[10] J. L. M. Benjamin, J. Curcio and P. Newman, “Navigation of unmanned marine vehicles in accordance with the rules of the road,” in *In proceedings of the Interational Conference on Robotics and Automation (IRCA) 2006*, 2006. [Online]. Available: http://oceanai.mit.edu/mikerb/publications/benjamin-icra-colregs-2006.pdf

[11] L. S. T. Bandyophadyay and F. Hover, “A simple reactive obstacle avoidance algorithm and its application in singapore harbor,” in *In proceedisng of the International Conference on Field and Service Robotics 2009*, 2009. [Online]. Available: http://censam.mit.edu/publications/tirtha09.pdf

[12] L. B. e. a. A. Viquerat, “Reactive collision avoidance for unmanned aerial vehicles using doppler radar,” in *In proceedings of the 6th International Conference on Field and Service Robotics - FSR 2007, Chamonix, France*, 2007. [Online]. Available: http://hal.archives-ouvertes.fr/docs/00/19/59/33/PDF/fsr_53.pdf